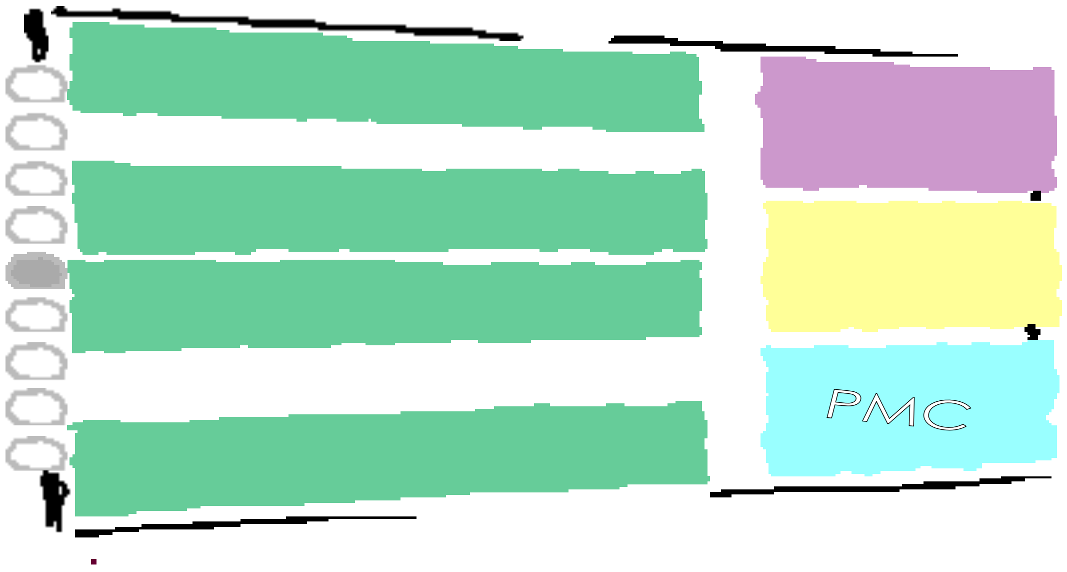
CMS Front-End Driver

PMC

Prototype

FED_Mon Manual

Version 2.0,January 2001



J. Coughlan
Rutherford Appleton Laboratory

1 Contents

2 Foreword

This documents describes the monitor program used for testing the CMS Front End Driver FED PMC when used on a CES RIO2 carrier processor board.

This document can be obtained from the following anonymous ftp site:

address : **ftp.te.rl.ac.uk**
directory: **cms/fed/fed_pmc/docs/manuals/ fedmon_um.pdf**

Please send any comments on the contents of this document to the author J.Coughlan@rl.ac.uk

3 Introduction

FED_Mon is a low level debugger & monitor program for the CMS Front-End Driver prototype FED-PMC[1] when running them on the CES RIO2 (including RTPC) family of VME PowerPC Single Board Computers [2].

FED_Mon is designed to provide a convenient way of standalone testing of FED-PMC modules in a test bench environment equipped with only a serial line and an ethernet connection to the RIO2.

This manual assumes the user is familiar with the RIO2 and the commands of it's associated monitor program PPC_Mon.

FED_Mon provides a means of taking simple data acquisition runs to test the operation of the FED-PMC. It also allows the user to reconfigure the firmware design of the FED-PMC Xilinx FPGA and provides several other low-level testing and debugging operations.

4 Getting Started

4.1 Getting a copy of the FED_Mon program

FED_Mon is a PowerPC XCOFF executable (text file) which can be found on the anonymous ftp server:
address : **ftp.te.rl.ac.uk**
directory: **cms/fed/fed_pmc/software/fedmon/**

4.2 Loading FED_Mon onto RIO2

The standard way of running FED_Mon is to load it from an tftp ("trivial ftp")[1] server into the RIO2 memory using the PPC_Mon "load" command.

The FED_Mon is XCOFF file relocated to run at the RIO2 system memory address $1'0000

NB The standard CES monitor PPC_Mon [3] must first be running in order to load FED_Mon from the server.

i) If you are already in FED_Mon use the "reset" command to return to PPC_Mon:

```
FED_Mon>reset
```

ii) First ensure the RIO2 host and tftpd server addresses are setup correctly:

```
PPC_Mon>show inet
```

These settings can be changed with the set command:

```
PPC_Mon>set inet_host your_RIO_IPaddr
PPC_Mon>set inet_server your_tftp_server_IPaddr
```

iii) Load FED_Mon at $1'0000:

```
PPC_Mon>load le fedmon_filename 10000
network boot
loading file 'fedmon' from server 130.246.40.163
ethernet local address = 00:80:a2:00:8a:33
ethernet address of server = 08:00:07:1e:2a:c5
loading at address:10000

tftp packet number:      200
file length = 0x46424 [287780]
transfer rate: 29 kbyte/sec
PPC_Mon>
```

Notes:

1. It is also possible to store the FED_Mon program permanently in the FLASH Memory of the RIO2. The RIO2 can then be configured to boot up with FED_Mon. See the RIO2 documentation for instructions.

Now FED_Mon is ready to run.

[1] tftp servers are available on most platforms.

4.3 Running FED_Mon

To run FED_Mon:

```
PPC_Mon>exec 10000

starting code execution at address: 4a614

FEDMON Version of Dec  6 2000 at 10:29:49

* WARNING => RUNNING FEDMON TEST VERSION.
PPC regs: msr  = 0x00001030; hid0 = 0x80000080;
INFO => DOING VME SETUP...
FED PMC @ VME A32 base ; len = 2 MB
SYS MEM start @ VME A32 base + $200000 ; len = 4 MB
RIO A24 base for CSR           = $xx460000
RIO A32 base for System Memory = $08000000

QUESTION => Which PMC Slot is FED in ? [1 (upper) / 2 (lower = default = CR)] =>
2
INFO => OK Using PMC Slot 2...
* WARNING => PCI CFG is using RIO2 MAPPING.
fedmem: pci_cfg_base = $80802000
fedmem: pci_mem_base_abs = $01200000
fedmem: pci_device_number = $03

bridge config regs @ pci cfg base = $80802000

            vendor id = $fed010dc (PCI9080_PCIIDR)
bridge cmd/status reg = $00000006 (PCI9080_PCICR)
     dpm pci mem base = $01200000 (PCI9080_PCIBAR2)
reg/fifo pci mem base = $01300000 (PCI9080_PCIBAR3)
  bridge pci mem base = $01300100 (PCI9080_PCIBAR0)
fedbrg: default values were set.

bridge local regs: @ pci mem base = $c1300100

      dpm local base = $00000001 (PCI9080_LAS0BA)
     dpm window size = $fff00000 (PCI9080_LAS0RR)
      dpm descriptor = $030300c3 (PCI9080_LBRD0)
     regs local base = $00100001 (PCI9080_LAS1BA)
    regs window size = $ffffff00 (PCI9080_LAS1RR)
     regs descriptor = $00000243 (PCI9080_LBRD1)
  mode & arbitration = $01200000 (PCI9080_MARBR)
        eeprom cntrl = $98000070 (PCI9080_CNTRL)
          big endian = $00000084 (PCI9080_BIGEND)

INFO => Xilinx FPGA status at start:
fedxlst: Xilinx Load status : INIT = 1, DONE = 0
INFO => Loading Xilinx from Flash EEPROM (takes approx 5 secs)...
INFO => End Xilinx load from Flash after 4559 msecs
fedxlst: Xilinx Load status : INIT = 1, DONE = 1
fedbrg: default values were set.

bridge local regs: @ pci mem base = $c1300100

      dpm local base = $00000001 (PCI9080_LAS0BA)
     dpm window size = $fff00000 (PCI9080_LAS0RR)
      dpm descriptor = $030300c3 (PCI9080_LBRD0)
     regs local base = $00100001 (PCI9080_LAS1BA)
    regs window size = $ffffff00 (PCI9080_LAS1RR)
     regs descriptor = $00000243 (PCI9080_LBRD1)
  mode & arbitration = $01200000 (PCI9080_MARBR)
        eeprom cntrl = $98020070 (PCI9080_CNTRL)
```

```
          big endian = $00000084 (PCI9080_BIGEND)

INFO => Setting up FED specific registers with defaults...
ATTENTION => If we stop here the Xilinx is NOT Loaded.

fedinit: using default values.
fedinit: WARNING => If Bus errors here, the CLOCK may NOT be running.
fedinit: WARNING => If we stop here the Xilinx may NOT be loaded.

fedstat: fed register status only:-

fedstat: ATTENTION => If we stop here the Xilinx is NOT Loaded.
fedstat: reset default bridge configuration.
fedstat: reset default bridge configuration.
 fed regs: @ pci mem base = $c1300080

 Digitisation is >> DISABLED <<

    Test mode is >> DISABLED <<

     Throttle is >> DISABLED <<
 throttle threshold = $000000ab

   Clock source = 0
   Clock source => PCI CLOCK
    Clock delay = 0

 Trigger source = 0
 Trigger source => FRONT PANEL TRIGGER
   Trigger mode = 0
   Trigger mode => START DIGITISATION

 adc_chan_mask = $aa
 ADC chans 0&1 are << ENABLED >>
           2&3 are << ENABLED >>
           4&5 are << ENABLED >>
           6&7 are << ENABLED >>
 ADC Sample Freq = 0
 ADC Sample Size = 16

 Number of Filled Buffers = 0

INFO => The FED PMC is Initialized and ready to take data.
INFO => FED Serial Number = 004 (dec)
fedrev:  Version information :

FIRMWARE:
Xilinx : version = 0
       : revison = 2
       : prototype = 0

SOFTWARE:
FED_Mon Version of Dec  6 2000 at 11:05:34
FEDPMC s/w library version = '2.07'
INFO => Type 'fed' for a full list of FED_Mon commands.
FED_Mon>
```

Notes:

1. FED_Mon begins by asking the user for the PMC slot number and then configures the FED-PMC PCI Bridge.

2. It then checks if the Xilinx FPGA is initialized and if not it tries to load it from the onboard Flash EEPROM (The Flash EEPROM is loaded prior to delivery with standard Xilinx configuration file).

NB Once the Xilinx is loaded the red LED which came on when power was applied should now go OFF (the Xilinx loading takes ≈ 5 seconds).

3. The FED-PMC registers are then initialized to some standard default values and their status displayed. Note that the FED-PMC serial number and the Xilinx FPGA firmware version are displayed.

The FED-PMC is now ready to take data.

5 Data taking

After startup FED_Mon initialises the FED-PMC with default settings which can be changed using appropiate commands. e.g. ADC sample size, clock source …etc

NB After intialisation the default settings use the internal PCI bus clock (33 MHz).

5.1 Running with the PCI clock and software triggers

IMPORTANT These data taking examples assume the FED-PMC FPGA is loaded with firmware version : v0r2 (i.e. old Scope mode).
For firmware version : v2rNN (i.e. Header Finding mode) additional commands are required.

The following example sends 4 software triggers and displays the start of the ADC contents for the first event only (in this example there are no connections to the data inputs hence the values ≈ 1/2 full scale).

i)
```
FED_Mon>fedstart 1
fedstart: FED data acquisition Run is STARTED, using SOFTWARE triggers.
```

ii)
```
FED_Mon>fedswt 4
fedswt: sent software trigger #1.
fedswt: sent software trigger #2.
fedswt: sent software trigger #3.
fedswt: sent software trigger #4.
fedbuf: number of filled buffers = 4.
```

iii)
```
FED_Mon>fedrout
readout time for 16 samples (skip begin/end = 0/0) = 452 micro sec
fedrout: Readout Event data to RIO address $00300000

Event no = $00000000 ; Bunch no = 49148
Event size = 256 bytes; Number Samples = 16; Skipped samples begin/end = 0/0;
Buffer base in dpm = $00000000

Unformatted data follows:

ADC Chan# =>  0    1    2    3    4    5    6    7
Sample #
       000:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
       001:  0284 0278 0274 0277 0280 0280 0281 0279   (Dec)
       002:  0284 0279 0274 0277 0280 0280 0280 0279   (Dec)
       003:  0284 0278 0274 0277 0280 0279 0280 0278   (Dec)
       004:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
       005:  0284 0278 0275 0277 0280 0280 0280 0279   (Dec)
       006:  0284 0279 0274 0277 0280 0280 0281 0279   (Dec)
       007:  0284 0279 0275 0277 0280 0280 0281 0279   (Dec)
       008:  0284 0279 0274 0277 0280 0280 0280 0279   (Dec)
       009:  0284 0279 0274 0277 0280 0280 0281 0279   (Dec)
       010:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
       011:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
       012:  0284 0278 0274 0277 0280 0280 0280 0278   (Dec)
       013:  0284 0278 0274 0277 0280 0280 0280 0278   (Dec)
       014:  0284 0279 0275 0277 0280 0280 0281 0279   (Dec)
       015:  0284 0279 0274 0277 0280 0280 0281 0279   (Dec)

Number of events pending = 3
FED_Mon>
```

iv)

```
FED_Mon>fedstop
fedstop: Event & BX Counters are RESET
fedstop: FED data acquisition Run is STOPPED.
FED_Mon>fedstop
```

6 Advanced Operations

6.1 Updating Firmware in Flash EEPROM

The FED-PMC firmware for the Xilinx FPGA is stored onboard in a Flash EEPROM.
The Flash device is loaded prior to delivery with the standard firmware design file.

Alternative firmware designs are made available from the anonymous ftp server:
address : **ftp.te.rl.ac.uk**
directory: **cms/fed/fed_pmc/firmware/**

The Flash contents can be updated with a new firmware file using the following procedure.
IMPORTANT: Updating the firmware design is not a common practice and may considerably alter the functional behaviour of the FED-PMC. If in doubt consult an expert before doing it.

Once the Flash is reprogrammed the FPGA firmware is permanently updated.

The procedure is as follows:

i) Reset the RIO2.

ii) First clear a block of memory for the new Xilinx file.

```
PPC_Mon>fm.w 100000..200000 0
PPC_Mon>
```

iii) Load the Xilinx file (rbt = raw bit file) from the tftp server into the cleared memory block.

```
PPC_Mon>load le fpga_filename 100000
network boot
loading file 'fpgav0r2p.rbt' from server 130.246.40.163
ethernet local address = 00:80:a2:00:8a:33
ethernet address of server = 08:00:07:1e:2a:c5
loading at address:100000

tftp packet number:      640
file length = 0xcbbb9 [834489]
transfer rate: 11 kbyte/sec
PPC_Mon>
```

iv) Reload and run FED_Mon.

```
PPC_Mon>exec 10000
starting code execution at address: 46930

...... more printout ......

 FIRMWARE:
 Xilinx : version = 0
        : revison = 2
        : prototype = 0

 SOFTWARE:
 FED_Mon Version of Dec  6 2000 at 11:05:34
 FEDPMC s/w library version = '2.07'
 INFO => Type 'fed' for a full list of FED_Mon commands.
FED_Mon>
```

v) Reload Flash EEPROM contents from file.
(NB The user is given a couple of chances to abort the action).

```
FED_Mon>fedldxfep
fedldxfep: default rio addresses xilinx file: source @ $00100000; dest @ $00200000
fedldxfep: WARNING: This command overwrites contents of Flash EEPROM with new
Xilinx file.
fedldxfep: Do you really want to do this?
fedldxfep: Type 'yes' to continue any other character to abort =>
fedldxfep: OK continuing...

nlines = 0 , xp[0] = $00200000 , total bytes = 0
nlines = 1 , xp[1] = $00200028 , total bytes = 40
nlines = 2 , xp[2] = $002001fd , total bytes = 509
nlines = 3 , xp[3] = $002003d2 , total bytes = 978
nlines = 4 , xp[4] = $002005a7 , total bytes = 1447
nlines = 5 , xp[5] = $0020077c , total bytes = 1916
nlines = 6 , xp[6] = $00200951 , total bytes = 2385
nlines = 7 , xp[7] = $00200b26 , total bytes = 2854
nlines = 8 , xp[8] = $00200cfb , total bytes = 3323
nlines = 9 , xp[9] = $00200ed0 , total bytes = 3792
nlines = 100 , xp[100] = $0020b587 , total bytes = 46471
nlines = 200 , xp[200] = $00216cbb , total bytes = 93371
nlines = 300 , xp[300] = $002223ef , total bytes = 140271
nlines = 400 , xp[400] = $0022db23 , total bytes = 187171
nlines = 500 , xp[500] = $00239257 , total bytes = 234071
nlines = 600 , xp[600] = $0024498b , total bytes = 280971
nlines = 700 , xp[700] = $002500bf , total bytes = 327871
nlines = 800 , xp[800] = $0025b7f3 , total bytes = 374771
nlines = 900 , xp[900] = $00266f27 , total bytes = 421671
nlines = 1000 , xp[1000] = $0027265b , total bytes = 468571
nlines = 1100 , xp[1100] = $0027dd8f , total bytes = 515471
nlines = 1200 , xp[1200] = $002894c3 , total bytes = 562371
nlines = 1300 , xp[1300] = $00294bf7 , total bytes = 609271
nlines = 1400 , xp[1400] = $002a032b , total bytes = 656171
nlines = 1500 , xp[1500] = $002aba5f , total bytes = 703071
nlines = 1600 , xp[1600] = $002b7193 , total bytes = 749971
nlines = 1700 , xp[1700] = $002c28c7 , total bytes = 796871
nlines = 1768 , xp[1768] = $002ca55b , total bytes = 828763
nlines = 1769 , xp[1769] = $002ca730 , total bytes = 829232
nlines = 1770 , xp[1770] = $002ca905 , total bytes = 829701
nlines = 1771 , xp[1771] = $002caada , total bytes = 830170
nlines = 1772 , xp[1772] = $002cacaf , total bytes = 830639
nlines = 1773 , xp[1773] = $002cae84 , total bytes = 831108
nlines = 1774 , xp[1774] = $002cb059 , total bytes = 831577
nlines = 1775 , xp[1775] = $002cb22e , total bytes = 832046
nlines = 1776 , xp[1776] = $002cb403 , total bytes = 832515
nlines = 1777 , xp[1777] = $002cb410 , total bytes = 832528
xilinx file,   total lines = 1777
xilinx file, header length = 40 bytes
xilinx file,  total length (including startup bits) = 832536 bytes
fedldxfep: xfile @ $00200000; nlines = 1777; bytes = 832536
fedldxfep: OK correct number of bits found for Xilinx load.
fedldxfep: bitstream_char_ptr = $00200000 ; bitstream_packed_ptr = $00300000
fedldxfep:  i  =  0  ;  *bitstream_char_ptr  =  $00000001  ;  *bitstream_packed_ptr  =
$00000001
fedldxfep: i = 100000 ; *bitstream_char_ptr = $00000000 ; *bitstream_packed_ptr =
$ffffffff
fedldxfep: i = 200000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$00000000
fedldxfep: i = 300000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$ffffffb5
```

```
fedldxfep: i = 400000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$ffffffd7
fedldxfep: i = 500000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$ffffffff
fedldxfep: i = 600000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$ffffffff
fedldxfep: i = 700000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$00000000
fedldxfep: i = 800000 ; *bitstream_char_ptr = $00000001 ; *bitstream_packed_ptr =
$00000000
fedldxfep: Xilinx packed bit stream loaded @ $00300000 ; number packed bytes =
104068.
fedldxfep: WARNING: Do you really want to change the contents of Flash EEPROM?
fedldxfep: Type 'yes' to continue any other character to abort =>
fedldxfep: OK continuing...

fedldxfep: new number_flash_pages = 407.
Loading Flash > page# = 0 ; source = $00300000 ; size = $00000100.
Loading Flash > page# = 100 ; source = $00306400 ; size = $00000100.
Loading Flash > page# = 200 ; source = $0030c800 ; size = $00000100.
Loading Flash > page# = 300 ; source = $00312c00 ; size = $00000100.
Loading Flash > page# = 400 ; source = $00319000 ; size = $00000100.
fedldxfep: Flash EEPROM contents are now reloaded.  Power FED off and on and then
reload FPGA from Flash.
FED_Mon>
```

Notes:

1. After the first user response the Xilinx file length and format are checked for consistency.

2. After the second user response the Flash EEPROM is actually updated.

3. Immediately after reloading the Flash the FED-PMC should be powered OFF & ON (to ensure the FPGA is updated with the new design).

6.2 Loading FPGA directly from file

In unusual cirumstances (e.g. to bootstrap the FED-PMC during production) the user may wish to update the FPGA directly (i.e. without permanently changing the Flash EEPROM contents).

This is possible with the following procedure.

Load the Xilinx file into the RIO using the same procedure as described in section Updating Firmware

```
FED_Mon>fedldxf

fedldxf: default rio addresses xilinx file: source @ $00100000; dest @ $00200000
fedldxf: WARNING: This command reloads Xilinx
fedldxf: Do you really want to do this?
fedldxf: Type 'yes' to continue any other character to abort =>
fedldxf: OK continuing...

nlines = 0 , xp[0] = $00200000 , total bytes = 0
nlines = 1 , xp[1] = $00200028 , total bytes = 40
nlines = 2 , xp[2] = $002001fd , total bytes = 509
nlines = 3 , xp[3] = $002003d2 , total bytes = 978
nlines = 4 , xp[4] = $002005a7 , total bytes = 1447
nlines = 5 , xp[5] = $0020077c , total bytes = 1916
nlines = 6 , xp[6] = $00200951 , total bytes = 2385
nlines = 7 , xp[7] = $00200b26 , total bytes = 2854
nlines = 8 , xp[8] = $00200cfb , total bytes = 3323
nlines = 9 , xp[9] = $00200ed0 , total bytes = 3792
nlines = 100 , xp[100] = $0020b587 , total bytes = 46471
nlines = 200 , xp[200] = $00216cbb , total bytes = 93371
nlines = 300 , xp[300] = $002223ef , total bytes = 140271
nlines = 400 , xp[400] = $0022db23 , total bytes = 187171
nlines = 500 , xp[500] = $00239257 , total bytes = 234071
nlines = 600 , xp[600] = $0024498b , total bytes = 280971
nlines = 700 , xp[700] = $002500bf , total bytes = 327871
nlines = 800 , xp[800] = $0025b7f3 , total bytes = 374771
nlines = 900 , xp[900] = $00266f27 , total bytes = 421671
nlines = 1000 , xp[1000] = $0027265b , total bytes = 468571
nlines = 1100 , xp[1100] = $0027dd8f , total bytes = 515471
nlines = 1200 , xp[1200] = $002894c3 , total bytes = 562371
nlines = 1300 , xp[1300] = $00294bf7 , total bytes = 609271
nlines = 1400 , xp[1400] = $002a032b , total bytes = 656171
nlines = 1500 , xp[1500] = $002aba5f , total bytes = 703071
nlines = 1600 , xp[1600] = $002b7193 , total bytes = 749971
nlines = 1700 , xp[1700] = $002c28c7 , total bytes = 796871
nlines = 1768 , xp[1768] = $002ca55b , total bytes = 828763
nlines = 1769 , xp[1769] = $002ca730 , total bytes = 829232
nlines = 1770 , xp[1770] = $002ca905 , total bytes = 829701
nlines = 1771 , xp[1771] = $002caada , total bytes = 830170
nlines = 1772 , xp[1772] = $002cacaf , total bytes = 830639
nlines = 1773 , xp[1773] = $002cae84 , total bytes = 831108
nlines = 1774 , xp[1774] = $002cb059 , total bytes = 831577
nlines = 1775 , xp[1775] = $002cb22e , total bytes = 832046
nlines = 1776 , xp[1776] = $002cb403 , total bytes = 832515
nlines = 1777 , xp[1777] = $002cb410 , total bytes = 832528
xilinx file,   total lines = 1777
xilinx file, header length = 40 bytes
xilinx file,  total length (including startup bits) = 832536 bytes
fedldxf: xfile @ $00200000; nlines = 1777; bytes = 832536
fedldxf: OK correct number of bits found for Xilinx load.
fedldxf: before loading status, result = 0, INIT = 1, DONE = 1
fedldxf: loading xfile to fpga XC4036_XL...
fedldxf: wrote_xilinx_header, result = 0, INIT = 1, DONE = 0
fedldxf: Frame 1, INIT = 1, DONE = 0
fedldxf: Frame 101, INIT = 1, DONE = 0
fedldxf: Frame 201, INIT = 1, DONE = 0
fedldxf: Frame 301, INIT = 1, DONE = 0
fedldxf: Frame 401, INIT = 1, DONE = 0
fedldxf: Frame 501, INIT = 1, DONE = 0
fedldxf: Frame 601, INIT = 1, DONE = 0
fedldxf: Frame 701, INIT = 1, DONE = 0
fedldxf: Frame 801, INIT = 1, DONE = 0
fedldxf: Frame 901, INIT = 1, DONE = 0
fedldxf: Frame 1001, INIT = 1, DONE = 0
fedldxf: Frame 1101, INIT = 1, DONE = 0
fedldxf: Frame 1201, INIT = 1, DONE = 0
fedldxf: Frame 1301, INIT = 1, DONE = 0
fedldxf: Frame 1401, INIT = 1, DONE = 0
fedldxf: Frame 1501, INIT = 1, DONE = 0
fedldxf: Frame 1601, INIT = 1, DONE = 0
fedldxf: Frame 1701, INIT = 1, DONE = 0
fedldxf: write_xilinx_footer, result = 0, INIT = 1, DONE = 1
fedldxf: ...XC4036_XL loaded ok.
FED_Mon>
```

Notes:

1. After the FED-PMC is powered OFF & ON the FPGA will revert to the previous design stored in the Flash EEPROM.

7 Trouble shooting with FED_Mon

The Xilinx may fail to load if the Flash EEPROM is not correctly loaded.

```
PPC_Mon>exec 10000
starting code execution at address: 45f70

FEDMON Version of Nov 18 1998 at 14:23:50

INFO => RUNNING FEDMON TEST VERSION.
PPC regs: msr  = 0x00001030; hid0 = 0x8100c080;
INFO => SKIPPING VME SETUP
INFO => PMC Simulation is OFF

fedrev:  Version information :

FED_Mon Version of Nov 18 1998 at 14:23:26


INFO => Initialising FED PMC PCI Bridge with default settings...
fedmem: set bridge pci cfg @ abs pci mem base = $00000000

INFO => Xilinx FPGA status at start:
fedlatst: status register : INIT = 1, DONE = 0

INFO => Loading Xilinx from Flash EEPROM...
INFO => start timeout counter = 4 (timeout @ 50000)
...timeout counter = 10000
...timeout counter = 20000
...timeout counter = 30000
...timeout counter = 40000
...timeout counter = 50000
INFO => end timeout counter = 50001
ATTENTION => >>>> The XILINX is NOT yet loaded <<<<<.
FED_Mon>
```

Suggestion: Try powering OFF and ON and repeating.

Suggestin : Try reloading the Flash EEPROM (see section 6.1).

8 Using PPC_Mon commands

PPC_Mon is the default CES monitor debugger program which starts when the RIO2 is booted up. FED_Mon is based on PPC_Mon and some of the PPC_Mon commands are available within FED_Mon.

Some useful ones include:

dm.w, dp.w : display memory locations
fm.w, fp.w : fill memory locations
reset : to return to PPC_Mon

WARNING The following commands are NOT available within FED_Mon and should ONLY be called from PPC_Mon. Use the “reset” command to leave FED_Mon and return to PPC_Mon before using them.

set : to set RIO2 NVRAM parameters
show : to display RIO2 NVRAM parameters
load : to load programs to RIO2 memory
exec : to run programs in RIO2 memory
boot : run boot code

Please refer to the PPC_Mon manual [3] for details of these (and other) commands.

9 FED_Mon Commands

```
*******************
Standard FED_Mon commands valid for ALL Firmware versions:
*******************
fedadc:         Display/Set ADC registers
fedbrg:         FED PMC Bridge PCI LOCAL registers.
fedbuf:         Display number of occupied buffers in dpm.
fedbufp:        Purge remaining occupied buffers in dpm.
fedbug:         Display FED program variables (for debugging).
fedckcfg:       Enter/Exit FPGA Clock configuration mode.
fedclk:         Display/Set Clock source & delay.
fedctr:         Reset Event and Bunch Crossing counters.
feddig:         Display/Set Digitisation mode
feddp:          Read from FED addresses
feddpmfl:       Fill entire DPM with identical value.
fedfeprd:       Read from Flash EPROM.
fedfepwr:       Write to Flash EPROM.
fedfifo:        Read FED fifo contents
fedfp:          Fill FED addresses
fedidrd:        Read FED serial number.
fedidwr:        Store FED serial number.
fedinit:        Initialise FED registers.
fedlatrd:       Read bits from Lattice serial line.
fedlatwr:       Write a bit to Lattice serial line.
fedlatfep:      Send Lattice command to load fpga from flash eeprom.
fedldxf:        Load Xilinx .rbt file from rio memory to FPGA.
fedldxfep:      Load Xilinx .rbt file from rio memory to Flash EEPROM.
fedmap:         Display FED address map.
fedmem:         Initialise FED PMC Bridge PCI CFG registers.
fedplxeper:     Erase contents of PLX serial eprom.
fedplxepfl:     Fill contents of PLX serial eprom with default values.
fedplxeprd:     Read contents of PLX serial eprom.
fedplxepwr:     Write to an individual location in PLX serial eprom.
fedres1:        Reserved command don't use.
fedres2:        Reserved command don't use.
fedrev:         Display revision info of FED h/w and s/w.
fedrout:        Readout and display fed event.
fedscp:         Continuous read or write from/to FED (Nb old meaning of scope).
fedstart:       Start FED data acquisition run.
fedstat:        Display FED status.
fedstop:        Stop FED data acquisition run.
fedswt:         Send Software triggers.
fedtest:        Test FED DPM.
fedthr:         Display/Set Throttle mode
fedtmod:        Display/Set Test mode
fedtrg:         Display/Set Trigger source & mode
fedxlst:        Display Lattice serial line FPGA status.

*******************
Following FED_Mon commands are ONLY valid for Firmware version 2 (rev 34 or
later):
*******************
fedapvctrd:     Display APV frame and trigger counters
fedapvctrs:     Reset APV frame and trigger counters
fedapvmod:      Display APV Sync status
fedapvparm:     Display/Set APV timeout parameters
fedapvpov:      Display/Set APV Timeout parameters override mode
fedapvscp:      Display APV (NEW) Scope status
```

```
fedapvthr:      Display/Set APV Header thresholds
fedfend:        Display FrontEnd to DPM status.
fedintfifo:     Display Internal FIFO status.
fedtrgflt:      Display Trigger Filter status.
```

Notes:

1. All FED_Mon command parameters are assumed to be HEXADECIMAL values. To enter other types specify the appropiate prefix eg 0d10 for decimal value 10.

9.1 FED

Name

fed – display list of all FED specific commands

Syntax

fed

Parameters

none

Description

List all FED_Mon commands.

Example

List FED_Mon commands.

```
FED_Mon>fed

*******************
Standard FED_Mon commands valid for ALL Firmware versions:
*******************
fedadc:         Display/Set ADC registers
fedbrg:         FED PMC Bridge PCI LOCAL registers.
fedbuf:         Display number of occupied buffers in dpm.
fedbufp:        Purge remaining occupied buffers in dpm.
fedbug:         Display FED program variables (for debugging).
fedckcfg:       Enter/Exit FPGA Clock configuration mode.
fedclk:         Display/Set Clock source & delay.
fedctr:         Reset Event and Bunch Crossing counters.
feddig:         Display/Set Digitisation mode
feddp:          Read from FED addresses
feddpmfl:       Fill entire DPM with identical value.
fedfeprd:       Read from Flash EPROM.
fedfepwr:       Write to Flash EPROM.
fedfifo:        Read FED fifo contents
fedfp:          Fill FED addresses
fedidrd:        Read FED serial number.
fedidwr:        Store FED serial number.
fedinit:        Initialise FED registers.
fedlatrd:       Read bits from Lattice serial line.
fedlatwr:       Write a bit to Lattice serial line.
fedlatfep:      Send Lattice command to load fpga from flash eeprom.
fedldxf:        Load Xilinx .rbt file from rio memory to FPGA.
fedldxfep:      Load Xilinx .rbt file from rio memory to Flash EEPROM.
fedmap:         Display FED address map.
fedmem:         Initialise FED PMC Bridge PCI CFG registers.
fedplxeper:     Erase contents of PLX serial eprom.
fedplxepfl:     Fill contents of PLX serial eprom with default values.
fedplxeprd:     Read contents of PLX serial eprom.
fedplxepwr:     Write to an individual location in PLX serial eprom.
fedres1:        Reserved command don't use.
fedres2:        Reserved command don't use.
fedrev:         Display revision info of FED h/w and s/w.
fedrout:        Readout and display fed event.
fedscp:           Continuous read or write from/to FED (Nb old meaning
of scope).
```

```
fedstart:       Start FED data acquisition run.
fedstat:        Display FED status.
fedstop:        Stop FED data acquisition run.
fedswt:         Send Software triggers.
fedtest:        Test FED DPM.
fedthr:         Display/Set Throttle mode
fedtmod:        Display/Set Test mode
fedtrg:         Display/Set Trigger source & mode
fedxlst:        Display Lattice serial line FPGA status.

*******************
Following FED_Mon commands are ONLY valid for Firmware version 2 (rev
34 or later):
*******************
fedapvctrd:     Display APV frame and trigger counters
fedapvctrs:     Reset APV frame and trigger counters
fedapvmod:      Display APV Sync status
fedapvparm:     Display/Set APV timeout parameters
fedapvpov:      Display/Set APV Timeout parameters override mode
fedapvscp:      Display APV (NEW) Scope status
fedapvthr:      Display/Set APV Header thresholds
fedfend:        Display FrontEnd to DPM status.
fedintfifo:     Display Internal FIFO status.
fedtrgflt:      Display Trigger Filter status.


Type 'help <command>' for command syntax:

NB Type 'help' to list in addition all PPC_Mon commands:

Notes: each command will use standard defaults if no arguments are
passed.
argument default is Hex (prefix 0d.. to specify decimal).

FED_Mon>
```

9.2 FEDADC

Name

fedadc – set up FED ADC registers

Syntax

fedadc
fedadc <adc_chan_mask> <adc_sample_freq> <adc_sample_size>

Parameters

< adc_chan_mask > {$0..$ f}
< adc_sample_freq > {0d0..0d255}
< adc_sample_size > {0d16..0d256..0d32768}

Description

Initialises the FED ADC registers.
If no arguments are passed the status is returned.

adc_chan_mask:	[$0..$f] bit #0 = 1/0 enables / disables ADC pair 0 & 1
	bit #1 = 1/0 enables / disables ADC pair 2 & 3
	bit #2 = 1/0 enables / disables ADC pair 4 & 5
	bit #3 = 1/0 enables / disables ADC pair 6 & 7
adc_sample_freq:	0 = Sample at clock frequency
	1 = Sample at 1/2 clock frequency
event_size:	Number of samples to readout per event
	(allowed values = 2^n where 4<= n <= 15).

Example

Enable all ADC channels . Sample at the clock frequency.
Readout 256 samples per trigger.

```
FED_Mon>fedadc f 0 0d256

 adc_chan_mask = $aa
 adc chans 0&1 are << ENABLED <<
           2&3 are >> ENABLED <<
           4&5 are >> ENABLED <<
           6&7 are >> ENABLED <<
 adc sample freq = 0
 adc sample size = 256 (decimal)
FED_Mon>
```

See also

fedinit, fedstat

9.3 FEDBRG

Name

fedbrg – set up fed PMC bridge PCI local registers

Syntax

fedbrg
fedbrg <dpm_size> <dpm_base> <dpm_desc> <reg_size> <reg_base> <reg_desc>

Parameters

< dpm_size >	size of window for DPM space {$fff0'0000}
< dpm_base >	local base for DPM space {$0000'0001}
< dpm_desc >	descriptor for DPM space{$030300c3}
< reg _size >	size of window for register space {$ffff'ff00}
< reg _base >	local base for register space {$0010'0001}
< reg _desc >	descriptor for register space{$00000243}

Description

Initialises the PMC bridge local registers.
If no arguments are passed defaults are used.
It maps PCI to local bus addresses.
Note: User should NOT need to change defaults which are set on initialisation.
fedmem must have been called at least once before calling fedbrg.

Example

Sets default local addresses.

```
FED_Mon>fedbrg
fedbrg: default values were set.

 bridge local regs: @ pci mem base = $c0100100

      dpm local base = $00000001
     dpm window size = $fff00000
      dpm descriptor = $000301c3
     regs local base = $00100001
    regs window size = $ffffff00
     regs descriptor = $00000243
  mode & arbitration = $01200000
        eeprom cntrl = $88000000
          big endian = $00000004
FED_Mon>
```

See also

fedmem, fedstat

9.4 FEDBUF

Name

fedbuf – Display number of occupied buffers in DPM.

Syntax

fedbuf

Parameters

none

Description

Gives the number of occupied buffers i.e. events pending to readout from the DPM.

Nb Number of occupied buffers is reset to 0 at new run start.

Example

Display number of occupied buffers.

```
FED_Mon>fedbuf
fedbuf: number of occupied buffers (pending readout) = 0.
FED_Mon>
```

See also

9.5 FEDBUFP

Name

fedbufp – Purge unread occupied buffers in DPM.

Syntax

fedbufp

Parameters

none

Description

Purges the unread buffers pending in the DPM.
This is done by emptying the event FIFO.

Example

Purge occupied buffers in DPM e.g. at the end of a run.

```
FED_Mon>fedbufp
fedbufp: purging remaining occupied buffers in dpm...
fedbufp: started with 8 occupied buffers.
fedbufp: Purged 8 buffers.
fedbufp: finished with 0 occupied buffers.
FED_Mon>
```

See also

9.6 FEDBUG

Name

fedbug – display FED_Mon program variables for debugging

Syntax

fedbug

Parameters

none

Description

Just for expert debugging of FED_Mon commands.

Example

Display monitor debugging information.

```
FED_Mon>fedbug
fedbug: OK: FED PMC S/W Simulation mode is OFF

fedbug: g_pci_cfg_base = $80802000
fedbug: g_pci_mem_base = $c0000000
fedbug:      pci mem base dpm = $c0000000
fedbug:     pci mem base fifo = $c0100000
fedbug: pci mem base fed regs = $c0100080
fedbug:   pci mem base bridge = $c0100100
FED_Mon>
```

See also

fedmap

9.7 FEDCKCFG

Name

fedckcfg – enter/exit FED FPGA clock configuration mode

Syntax

fedckcfg [<operation>]

Parameters

< operation>

0	Enter clock configuration mode
1	Exit clock configuration mode

Description

Command for Experts only.
NB Entering Clock Configuration mode **resets** Xilinx to default settings.
Therefore user should reinitialise FED settings with fedinit afterwards.

Example

Enter clock configuration mode.

```
FED_Mon>fedckcfg 1
fedckcfg: ENTER FPGA Clock configuration mode.
FED_Mon>
```

See also

fedclk, fedinit

9.8 FEDCLK

Name

fedclk – display/set FED clock source and delay

Syntax

fedclk
fedclk <clock_source><clock_delay>

Parameters

< clock_source > {0=PCI;2=Front Panel;3=J4 LVDS}
< clock_delay> {0d0..0d10}

Description

Initialises the FED clock source and delay.
Actual delay is ≈ <clock_delay> x 2.5 nsec. (i.e. 0 – 25 nsec range).
If no arguments are passed the clock status is displayed.
WARNING If the selected clock is not present (e.g. front panel clock) subsequent card access will return bus errors. Card may then require a hardware reset and reinitialisation in order to continue.

Example

Use PCI clock. Set clock skew delay to 10 nsec (4 x 2.5 nsec).

```
FED_Mon>fedclk 0 4
fedclk: clock source = PCI CLOCK
fedclk:  clock delay = 4 (x 2.5 nsec)
fedclk: WARNING => If Bus errors after changing, new CLOCK may NOT be
running.
FED_Mon>
```

See also

fedckcfg, fedinit

9.9 FEDCTR

Name

fedctr – Reset Event and Bunch crossing counters.

Syntax

fedctr

Parameters

none

Description

Resets the Event and Bunch crossing counters.
(Switches in and out of test mode if necessary).
NB Counters are read from Event FIFO, i.e. during readout process.

Example

Reset counters (e.g. before start of new run).

```
FED_Mon>fedctr
fedctr: event & bunch counters are now reset.
FED_Mon>
```

See also

9.10 FEDDIG

Name

feddig – display/set FED Digitisation mode

Syntax

feddig
feddig [<operation>]

Parameters

< operation>

0	Disable ADC Digitisation mode
1	Enable ADC Digitisation mode

Description

Sets/Displays ADC Digitisation mode status.
If no argument is passed the status is displayed.
Enabling digitisation resets Buffer management logic.
New buffers are filled starting from base of DPM.
Note: Event FIFO contents are not cleared by this command. Use fedbufp.

Example

Start ADC digitisation.

```
FED_Mon>feddig 1
feddig: Digitisation mode is << ENABLED >>
FED_Mon>
```

See also

9.11 FEDDP

Name

feddp – display FED address contents

Syntax

feddp
feddp <address_offset><number_reads>

Parameters

< address_offset > offset from FED memory base
< number_reads > number of longs to read & display

Description

Displays memory contents starting from base of DPM.
If no arguments are passed the first location in the DPM is displayed.

Example

Display first 4 long words in DPM.

```
FED_Mon>feddp 0 4
read: op #000 @ address = $c0000000 : data = $01030111
read: op #001 @ address = $c0000004 : data = $01100110
read: op #002 @ address = $c0000008 : data = $010e010f
read: op #003 @ address = $c000000c : data = $0110010d
FED_Mon>
```

See also

fedfp, fedscp

and PPC_Mon command dm

9.12 FEDDPMFL

Name

feddpmfl – Fill entire DPM with identical value.

Syntax

feddpmfl <data>

Parameters

< data> 32-bit data value to fill in all locations of dpm

Description

Fill entire DPM with identical value.

Example

Fill entire DPM with$beefface.

```
FED_Mon>feddpmfl beefface
feddpmfl:  Fill entire DPM with identical value.

feddpmfl: DPM filled everywhere with $beefface
FED_Mon>

check new contents of dpm using PPC_Mon dp command
(dpm base address obtained with fedmap)

FED_Mon>dp.w c1200000
0xc1200000 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200010 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200020 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200030 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200040 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200050 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200060 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
0xc1200070 : 00ef00ce 00ef00ce 00ef00ce 00ef00ce  ................
FED_Mon>
```

See also

fedfp, feddp

and PPC_Mon commands dm, dp, fm, fp

9.13 FEDFEPRD

Name

fedfeprd – read contents of Flash EEPROM memory.

Syntax

fedfeprd
fedfeprd<dest_address> <number_pages><skip_flag>

Parameters

<dest_address > in RIO system mempty {$50’0000}
< number_pages >number of Flash EEPROM pages to dump { 1 }
< skip_flag >skip bits in memory not used by Xilinx file { 0}

Description

Dumps the contents of the Flash EEPROM memory to a specified address in RIO memory.
Note: Page size = 264 bytes.
If skip flag is set only the first 256 bytes of each page (used by Xilinx file) are dumped.
If no arguments are passed defaults are used.

Example

Display contents of first page in Flash EEPROM.

```
FED_Mon>fedfeprd
fedfeprd:  Read Flash EEPROM contents: dest @ $00500000; npages = 1
FEDPMC_DEBUG > fedpmc_fpga_eeprom_read_memory() : npages = 1 ; dest =
$00500000.
total flash reads = 2113 ; writes (cmds) = 64 ; clocks = 2177
FED_Mon>
FED_Mon>dm.w 500000
0x00500000 : ff20cb40 9f5bffdf ddfddfdd fddfddfd  ................
0x00500010 : af7afddf db7f77f7 7f77f77f 6bdebdeb  .z....w..w..k...
0x00500020 : debdfebf ebdebdeb deffeffe bfebdefd  ................
0x00500030 : eaf7affb f7bffbff bdfafffd ddfdffeb  ................
0x00500040 : bffdffdf fdffdffd ffdfffff ff7dfff7  ................
0x00500050 : ff7ff7ff 7ff7ffed 7ed7ed7e dffed7ed  ................
0x00500060 : fedfedfe d7ed7ed7 edfedffb 7fb5fb7f  ................
0x00500070 : 35f35f5d ffdddfdf fffe5fff ffffffff  5...............
FED_Mon>
```

See also

fedfepwr, fedldxfep

9.14 FEDFEPWR

Name

fedfepwr – write to Flash EEPROM memory.

Syntax

fedfepwr
fedfepwr<source_address> <number_pages>

Parameters

<source_address > in RIO system mempty {$40’0000}
< number_pages >number of Flash EEPROM pages to fill { 1 }

Description

WARNING: This command is for Experts only.

Fill Flash EEPROM memory from contents at a specified address in RIO memory.
Note: Page size = 264 bytes.
If no arguments are passed defaults are used.

Flash EEPROM is preloaded at delivery with standard FED Xilinx configuration file.
To load new configuration file use fedldxfep.
Writing to the Flash EEPROM will overwrite the existing Xilinx file in memory.

Example

Overwrite first page in Flash EEPROM.

```
FED_Mon>fedfepwr
fedfepwr: This command overwrites contents of Flash EEPROM
fedfepwr: Do you really want to do this?
fedfepwr: Type 'yes' to continue any other character to abort =>yes
fedfepwr: OK continuing...
fedfepwr:  Write to Flash EEPROM: source @ $00400000; npages = 1
FEDPMC_DEBUG  >  fedpmc_fpga_eeprom_load()  :  npages  =  1  ;  source  =
$00400000.
FEDPMC_DEBUG > fedpmc_fpga_eeprom_write_memory() : npages = 1 ; source
= $00400000.
FEDPMC_DEBUG > page = 0 ; source = $00400000 ; size = $00000100.
total flash writes = 2144 ; clocks = 4257
FED_Mon>
```

See also

fedfeprd, fedldxfep

9.15 FEDFIFO

Name

fedfifo – display contents of Event FIFO

Syntax

fedfifo <number_reads> { 1 }

Parameters

< number_reads >number of reads from Event FIFO

Description

Reads a specified number of words from the FIFO.
If no arguments are passed a single word is read.
Note: This command is only intended for debugging purposes
(extra reads from the FIFO will disrupt the normal readout routine sequence).

Example

Read 6 words from fifo.

```
FED_Mon>fedfifo 6
fifo read #1 : @ $00000000 : contents = $0003000f
fifo read #2 : @ $00000000 : contents = $00030000
fifo read #3 : @ $00000000 : contents = $00035533
fifo read #4 : @ $00000000 : contents = $0003010f
fifo read #5 : @ $00000000 : contents = $00030100
fifo read #6 : @ $00000000 : contents = $00039019
fedfifo: read fifo ; 6 times.
FED_Mon>
```

See also

fedrout

9.16 FEDFP

Name

fedfp – fills FED address contents with a fixed pattern.

Syntax

fedfp
fedfp <address_offset><number_fills>

Parameters

< address_offset > offset from FED memory base { $0 }
< number_reads > number of longs to fill { 1 }

Description

Fills memory contents starting from base of DPM with a fixed pattern.
If no arguments are passed the first location in the DPM is filled.
NB ADC Digitisation must be **disabled** in order to write new values to DPM.

Example

Fill first 4 long words in DPM and display updated contents.

```
FED_Mon>feddig 0
feddig: set Digitisation mode DISABLED.

FED_Mon>fedfp 0 4

FED_Mon>feddp 0 4
read: op #000 @ address = $c0000000 : data = $00000001
read: op #001 @ address = $c0000004 : data = $00000002
read: op #002 @ address = $c0000008 : data = $00000003
read: op #003 @ address = $c000000c : data = $00000004
FED_Mon>
```

See also

feddp, fedscp, feddpmfl

9.17 FEDIDRD

Name

fedidrd – Read FED serial number.

Syntax

fedidrd

Parameters

none

Description

Reads back the FED-PMC serial number stored in the Flash EEPROM.

Example

Display FED serial number.

```
FED_Mon>fedidrd
INFO => FED Serial Number = 004 (dec)
FED_Mon>
```

See also

fedidwr, fedrev

9.18 FEDIDWR

Name

fedidwr – Store FED serial number.

Syntax

fedidwr <serial_nr>

Parameters

< serial_nr> number corresponding to label on PMC

Description

WARNING: This command is for Experts only.

Stores the FED-PMC serial number (labelled on PMC) in the Flash EEPROM.

Example

Store FED serial number.

```
FED_Mon>fedidwr 004
fedidwr: WARNING: This command overwrites FED serial number in Flash
EEPROM
fedidwr: Do you really want to do this?
fedidwr: Type 'yes' to continue any other character to abort =>
fedidwr: OK continuing...
fedidwr: Write Serial Number afed0004 to Flash EEPROM
total flash writes = 224 ; clocks = 1731
FED_Mon>
```

Check the value

```
FED_Mon>fedidrd
INFO => FED Serial Number = 004 (dec)
FED_Mon>
```

See also

fedidrd

9.19 FEDINIT

Name

fedinit – set up FED specific registers

Syntax

```
fedinit
fedinit  <clock_source><clock_delay> <trigger_source> <trigger_mode>
           <adc_chan_mask><adc_sample_freq><adc_sample_size>
            <trigger_throttle_enable><trigger_throttle_threshold>
```

Parameters

< clock_source > {0=PCI;2=Front Panel;3=J4 LVDS}
< clock_delay> {0d0..0d10}
< trigger_source > {0=Front Panel;2=J4 LVDS;3=J4 TTL}
< trigger_mode > {0=Start Digitisation}
< adc_chan_mask > {$0..$f}
< adc_sample_freq > {$0..$ff}
< adc_sample_size > {0d16..0d256..0d32768}
< trigger_throttle_enable > {0=OFF;1=ON}
< trigger_throttle_threshold > {$0..$3ff}

Description

Initialises the FED specific registers.
If no arguments are passed defaults are used.

Actual clock delay is ≈ <clock_delay> x 2.5 nsec. (i.e. 0 – 25 nsec range).

adc_chan_mask:	[$0..$f] bit #0 = 1/0 enables / disables ADC pair 0 & 1
	bit #1 = 1/0 enables / disables ADC pair 2 & 3
	bit #2 = 1/0 enables / disables ADC pair 4 & 5
	bit #3 = 1/0 enables / disables ADC pair 6 & 7
adc_sample_freq:	Number of clock cycles between samples.
event_size:	Number of samples to readout per event. (granularity = 16 samples/event).

Example

Initialise FED PMC with defaults.

```
FED_Mon>fedinit

fedinit: using default values.
fedinit: WARNING => If Bus errors here, the CLOCK may NOT be running.
fedinit: WARNING => If we stop here the Xilinx may NOT be loaded.

fedstat: fed register status only:-

fedstat: ATTENTION => If we stop here the Xilinx is NOT Loaded.
fedstat: reset default bridge configuration.
fedstat: reset default bridge configuration.
 fed regs: @ pci mem base = $c0100080

 Digitisation is >> DISABLED <<

    Test mode is >> DISABLED <<

     Throttle is >> DISABLED <<
 throttle threshold = $000000ab

   Clock source = 0
   Clock source => PCI CLOCK
    Clock delay = 0

 Trigger source = 0
 Trigger source => FRONT PANEL TRIGGER
   Trigger mode = 0
   Trigger mode => START DIGITISATION

 adc_chan_mask = $aa
 ADC chans 0&1 are << ENABLED >>
           2&3 are << ENABLED >>
           4&5 are << ENABLED >>
           6&7 are << ENABLED >>
 ADC Sample Freq = 0
 ADC Sample Size = 256

 Number of Filled Buffers = 0

FED_Mon>
```

See also

fedstat, fedmem, fedbrg

9.20 FEDLATFEP

Name

fedlatfep – Send Lattice command to load Xilinx from Flash EEPROM.

Syntax

fedlatfep

Parameters

none

Description

Sends the Lattice command to load Xilinx from Flash EEPROM.

WARNING: This command is for Experts only.
Misuse can disrupt operation of card, necessitating a hardware reset to recover.
E.g. should only be called if Xilinx is not already loaded.

Example

Load Xilinx from Flash EEPROM.

```
FED_Mon>fedlatfep
fedlatfep: send Lattice command to load fpga from flash eeprom
FED_Mon>
```

See also

9.21 FEDLATRD

Name

fedlatrd – Read bits via serial line connecting PCI bridge to Lattice™ CPLD.

Syntax

fedlatrd
fedlatrd <number_reads>

Parameters

< number_reads > number of bits to read.

Description

Reads and displays a specified number of bits via the serial line.
If no arguments are passed one bit is read.

WARNING: This command is for Experts only.
Misuse can disrupt operation of card, necessitating a hardware reset to recover.

Example

Read 4 bits from Lattice via serial line.

```
FED_Mon>fedlatrd 4
<<<< fedlatrd: read from lattice serial line : 4 times
read: i = 0; data = 0
read: i = 1; data = 0
read: i = 2; data = 0
read: i = 3; data = 0
total lattice reads = 260 ; clocks = 432
FED_Mon>
```

See also

fedlatwr

9.22 FEDLATWR

Name

fedlatwr – Send a single bit via serial line connecting PCI bridge to Lattice™ CPLD.

Syntax

fedlatwr
fedlatwr <bit_value>

Parameters

< bit_value> = 0/1.

Description

WARNING: This command is for Experts only.
Misuse can disrupt operation of card, necessitating a hardware reset to recover.

Send a single bit of data via the serial line.
If no arguments are passed 0 is sent.

Example

Send a '1' to Lattice via serial line.

```
FED_Mon>fedlatwr 1
FED_Mon>
```

See also

fedlatrd

9.23 FEDLDXF

Name

fedldxf – load Xilinx configuration file directly to FPGA

Syntax

fedldxf
fedldxf <source_address> <dest_address>

Parameters

< source_address > seen from RIO {$10’0000}
< dest_address > seen from RIO {$20’0000}

Description

The fedldxf command takes a Xilinx “rbt” formatted configuration file preloaded at <source_address> in system memory, verifies it and writes out formatted data for FPGA to < dest_address >. Finally it loads FPGA with formatted data.
If no arguments are passed defaults are used.
Care should be taken to ensure there is sufficient space above each address chosen (depending on Xilinx file size) if the defaults are not used (e.g. 1 Mbyte for standard XC4036XL .rbt file).

NB: Xilinx configuration file must be already loaded in memory before calling fedldxf.

See section Updating Firmware.

Example

Load Xilinx from configuration file preloaded in memory @ $10’0000.

```
FED_Mon>fedldxf
fedldxf: default rio addresses xilinx file: source @ $00100000; dest @
$00200000
 fedldxf: This command reloads Xilinx
 fedldxf: Do you really want to do this?
 fedldxf: Type 'yes' to continue any other character to abort =>
 fedldxf: OK continuing...

nlines = 0 , xp[0] = $00200000 , total bytes = 0
nlines = 1 , xp[1] = $00200028 , total bytes = 40
nlines = 2 , xp[2] = $002001fd , total bytes = 509
nlines = 3 , xp[3] = $002003d2 , total bytes = 978
nlines = 4 , xp[4] = $002005a7 , total bytes = 1447
nlines = 5 , xp[5] = $0020077c , total bytes = 1916
nlines = 6 , xp[6] = $00200951 , total bytes = 2385
nlines = 7 , xp[7] = $00200b26 , total bytes = 2854
nlines = 8 , xp[8] = $00200cfb , total bytes = 3323
nlines = 9 , xp[9] = $00200ed0 , total bytes = 3792
nlines = 100 , xp[100] = $0020b587 , total bytes = 46471
nlines = 200 , xp[200] = $00216cbb , total bytes = 93371
nlines = 300 , xp[300] = $002223ef , total bytes = 140271
nlines = 400 , xp[400] = $0022db23 , total bytes = 187171
nlines = 500 , xp[500] = $00239257 , total bytes = 234071
nlines = 600 , xp[600] = $0024498b , total bytes = 280971
nlines = 700 , xp[700] = $002500bf , total bytes = 327871
nlines = 800 , xp[800] = $0025b7f3 , total bytes = 374771
nlines = 900 , xp[900] = $00266f27 , total bytes = 421671
nlines = 1000 , xp[1000] = $0027265b , total bytes = 468571
nlines = 1100 , xp[1100] = $0027dd8f , total bytes = 515471
nlines = 1200 , xp[1200] = $002894c3 , total bytes = 562371
nlines = 1300 , xp[1300] = $00294bf7 , total bytes = 609271
nlines = 1400 , xp[1400] = $002a032b , total bytes = 656171
nlines = 1500 , xp[1500] = $002aba5f , total bytes = 703071
nlines = 1600 , xp[1600] = $002b7193 , total bytes = 749971
nlines = 1700 , xp[1700] = $002c28c7 , total bytes = 796871
nlines = 1768 , xp[1768] = $002ca55b , total bytes = 828763
nlines = 1769 , xp[1769] = $002ca730 , total bytes = 829232
nlines = 1770 , xp[1770] = $002ca905 , total bytes = 829701
nlines = 1771 , xp[1771] = $002caada , total bytes = 830170
nlines = 1772 , xp[1772] = $002cacaf , total bytes = 830639
nlines = 1773 , xp[1773] = $002cae84 , total bytes = 831108
nlines = 1774 , xp[1774] = $002cb059 , total bytes = 831577
nlines = 1775 , xp[1775] = $002cb22e , total bytes = 832046
nlines = 1776 , xp[1776] = $002cb403 , total bytes = 832515
nlines = 1777 , xp[1777] = $002cb410 , total bytes = 832528
xilinx file,   total lines = 1777
xilinx file, header length = 40 bytes
xilinx file,  total length (including startup bits) = 832536 bytes
fedldxf: xfile @ $00200000; nlines = 1777; bytes = 832536
fedldxf: OK correct number of bits found for Xilinx load.
fedldxf: before loading status, result = 0, INIT = 1, DONE = 1
fedldxf: loading xfile to fpga XC4036_XL...
fedldxf: wrote_xilinx_header, result = 0, INIT = 1, DONE = 0
fedldxf: Frame 1, INIT = 1, DONE = 0
fedldxf: Frame 101, INIT = 1, DONE = 0
fedldxf: Frame 201, INIT = 1, DONE = 0
fedldxf: Frame 301, INIT = 1, DONE = 0
fedldxf: Frame 401, INIT = 1, DONE = 0
fedldxf: Frame 501, INIT = 1, DONE = 0
fedldxf: Frame 601, INIT = 1, DONE = 0
fedldxf: Frame 701, INIT = 1, DONE = 0
fedldxf: Frame 801, INIT = 1, DONE = 0
fedldxf: Frame 901, INIT = 1, DONE = 0
fedldxf: Frame 1001, INIT = 1, DONE = 0
fedldxf: Frame 1101, INIT = 1, DONE = 0
fedldxf: Frame 1201, INIT = 1, DONE = 0
fedldxf: Frame 1301, INIT = 1, DONE = 0
fedldxf: Frame 1401, INIT = 1, DONE = 0
fedldxf: Frame 1501, INIT = 1, DONE = 0
fedldxf: Frame 1601, INIT = 1, DONE = 0
fedldxf: Frame 1701, INIT = 1, DONE = 0
fedldxf: write_xilinx_footer, result = 0, INIT = 1, DONE = 1
fedldxf: ...XC4036_XL loaded ok.
FED_Mon>
```

See also

fedldxfep

9.24 FEDLDXFEP

Name

fedldxfep – load Xilinx configuration file to Flash EEPROM

Syntax

fedldxfep
fedldxfep <source_address> <dest_address><packed_address>

Parameters

< source_address > seen from RIO {$10'0000}
< dest_address > seen from RIO {$20'0000}
< packed_address > seen from RIO {$30'0000}

Description

WARNING: This command is for Experts only. It PERMANENTLY changes the FED-PMC firmware.

The fedldxf command takes a Xilinx "rbt" formatted configuration file preloaded at <source_address> in system memory, verifies it and writes out formatted data for Flash EEPROM to < packed_address >. Finally it loads Flash EEPROM with formatted data.
If no arguments are passed defaults are used.
Care should be taken to ensure there is sufficient space above each address chosen (depending on Xilinx file size) if the defaults are not used (e.g. 1 Mbyte for standard XC4036XL .rbt file).

NB: Xilinx configuration file must be already loaded in memory before calling fedldxf.

Flash EEPROM is preloaded at delivery with standard FED Xilinx configuration file.

See section Loading FPGA directly from file.

Example

Load Flash EEPROM from configuration file preloaded in memory @ $10'0000.

```
FED_Mon>fedldxfep
fedldxfep: default rio addresses xilinx file: source @ $00100000; dest
@ $00200000
fedldxfep: This command overwrites contents of Flash EEPROM with new
Xilinx file.
fedldxfep: Do you really want to do this?
fedldxfep: Type 'yes' to continue any other character to abort =>yes

nlines = 0 , xp[0] = $00200000 , total bytes = 0
nlines = 1 , xp[1] = $00200028 , total bytes = 40
nlines = 2 , xp[2] = $002001fd , total bytes = 509
nlines = 3 , xp[3] = $002003d2 , total bytes = 978
nlines = 4 , xp[4] = $002005a7 , total bytes = 1447
nlines = 5 , xp[5] = $0020077c , total bytes = 1916
nlines = 6 , xp[6] = $00200951 , total bytes = 2385
nlines = 7 , xp[7] = $00200b26 , total bytes = 2854
nlines = 8 , xp[8] = $00200cfb , total bytes = 3323
nlines = 9 , xp[9] = $00200ed0 , total bytes = 3792
nlines = 100 , xp[100] = $0020b587 , total bytes = 46471
nlines = 200 , xp[200] = $00216cbb , total bytes = 93371
nlines = 300 , xp[300] = $002223ef , total bytes = 140271
nlines = 400 , xp[400] = $0022db23 , total bytes = 187171
nlines = 500 , xp[500] = $00239257 , total bytes = 234071
nlines = 600 , xp[600] = $0024498b , total bytes = 280971
nlines = 700 , xp[700] = $002500bf , total bytes = 327871
nlines = 800 , xp[800] = $0025b7f3 , total bytes = 374771
nlines = 900 , xp[900] = $00266f27 , total bytes = 421671
nlines = 1000 , xp[1000] = $0027265b , total bytes = 468571
nlines = 1100 , xp[1100] = $0027dd8f , total bytes = 515471
nlines = 1200 , xp[1200] = $002894c3 , total bytes = 562371
nlines = 1300 , xp[1300] = $00294bf7 , total bytes = 609271
nlines = 1400 , xp[1400] = $002a032b , total bytes = 656171
nlines = 1500 , xp[1500] = $002aba5f , total bytes = 703071
nlines = 1600 , xp[1600] = $002b7193 , total bytes = 749971
nlines = 1700 , xp[1700] = $002c28c7 , total bytes = 796871
nlines = 1768 , xp[1768] = $002ca55b , total bytes = 828763
nlines = 1769 , xp[1769] = $002ca730 , total bytes = 829232
nlines = 1770 , xp[1770] = $002ca905 , total bytes = 829701
nlines = 1771 , xp[1771] = $002caada , total bytes = 830170
nlines = 1772 , xp[1772] = $002cacaf , total bytes = 830639
nlines = 1773 , xp[1773] = $002cae84 , total bytes = 831108
nlines = 1774 , xp[1774] = $002cb059 , total bytes = 831577
nlines = 1775 , xp[1775] = $002cb22e , total bytes = 832046
nlines = 1776 , xp[1776] = $002cb403 , total bytes = 832515
nlines = 1777 , xp[1777] = $002cb410 , total bytes = 832528
xilinx file,   total lines = 1777
xilinx file, header length = 40 bytes
xilinx file,  total length (including startup bits) = 832536 bytes
fedldxfep: xfile @ $00200000; nlines = 1777; bytes = 832536
fedldxfep: OK correct number of bits found for Xilinx load.
fedldxfep:  bitstream_char_ptr  =  $00200000  ;  bitstream_packed_ptr  =
$00300000
fedldxfep:   i   =   0   ;   *bitstream_char_ptr   =   $00000001   ;
*bitstream_packed_ptr = $ffffffff
fedldxfep:  i  =  100000  ;  *bitstream_char_ptr  =  $00000000  ;
*bitstream_packed_ptr = $00000000
fedldxfep:  i  =  200000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $ffffffff
fedldxfep:  i  =  300000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $00000000
fedldxfep:  i  =  400000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $fffffff9
fedldxfep:  i  =  500000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $ffffffff
fedldxfep:  i  =  600000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $00000000
fedldxfep:  i  =  700000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $00000000
fedldxfep:  i  =  800000  ;  *bitstream_char_ptr  =  $00000001  ;
*bitstream_packed_ptr = $ffffffff
fedldxfep: Xilinx packed bit stream loaded @ $00300000 ; number packed
bytes = 104068.
FEDPMC_XILINX_TRACE:  Load  packed  bytes  to  flash  eeprom  ;  type  any
character to continue...
fedldxfep: new number_flash_pages = 407.
FEDPMC_DEBUG  >  fedpmc_fpga_eeprom_load()  :  npages  =  407  ;  source  =
$00300000.
FEDPMC_DEBUG  >  fedpmc_fpga_eeprom_write_memory()  :  npages  =  407  ;
source = $00300000.
FEDPMC_DEBUG > page = 0 ; source = $00300000 ; size = $00000100.
FEDPMC_DEBUG > page = 100 ; source = $00306400 ; size = $00000100.
FEDPMC_DEBUG > page = 200 ; source = $0030c800 ; size = $00000100.
FEDPMC_DEBUG > page = 300 ; source = $00312c00 ; size = $00000100.
FEDPMC_DEBUG > page = 400 ; source = $00319000 ; size = $00000100.
FED_Mon>
```

See also

fedldxf

9.25 FEDMAP

Name

fedmap – display FED address map

Syntax

fedmap

Parameters

none

Description

Just for expert debugging of register contents.
Useful if used with PPC_MON “dm & dp” commands.

Nb Address values may vary according to PMC slot number and FED_Mon version.

Example

Display FED base addresses.

```
FED_Mon>fedmap
fedmap:  Memory Map for FED PMC :

      pci mem base dpm = $c0000000
     pci mem base fifo = $c0100000
 pci mem base fed regs = $c0100080
   pci mem base bridge = $c0100100
FED_Mon>
```

then to display fed regs status:

```
FED_Mon>dp.w c1000080
0xc1000080 : 0000aa00 000000ab 00000100 0000000f  ................
0xc1000090 : 00000000 000003ff 000003ff 00000000  ................
0xc10000a0 : 00100000 00000000 00000000 000000ff  ................
0xc10000b0 : 000000ff 000000ff 001000b8 001000bc  ................
0xc10000c0 : 001000c0 001000c4 001000c8 001000cc  ................
0xc10000d0 : 001000d0 001000d4 001000d8 001000dc  ................
0xc10000e0 : 001000e0 001000e4 001000e8 001000ec  ................
0xc10000f0 : 001000f0 001000f4 00000001 fed00245  ...............E
FED_Mon>
```

See also

fedbug

9.26 FEDMEM

Name

fedmem – set up FED PMC bridge PCI configuration registers

Syntax

fedmem
fedmem <pci_cfg_base> <pci_mem_base>

Parameters

<pci_cfg_base> PCI configuration base address of PMC seen from RIO {$8080’2000}
<pci_mem_base> PCI memory base address for FED seen from RIO {$c000’0000}

Description

Initialise the PMC bridge configuration registers.
If no arguments are passed defaults are used.
Default settings assume FED is on RIO PMC lower slot no 1.
Note: User should NOT need to change defaults which are set on initialisation.

Example

Use default settings.

```
FED_Mon>fedmem
fedmem: default values were set.

bridge config regs @ pci cfg base = $80802000

                 vendor id = $908010b5
      bridge cmd/status reg = $02800007
     dpm pci mem base (rio) = $c0000000
    fifo pci mem base (rio) = $c0100000
  bridge pci mem base (rio) = $c0100100
FED_Mon>
```

See also

fedbrg, fedstat

9.27 FEDPLXEPER

Name

fedplxeper: Erase contents of PLX serial eprom.

Syntax

fedplxeper

Parameters

none

Description

WARNING: This command is for Experts only.

Erases the contents of serial eprom attached to PLX PCI bridge chip.
After erasing eprom contents readback as 0xFFFF.

Example

Erase PLX eprom contents.

```
FED_Mon>fedplxeper
fedplxeper:  Erase contents of PLX serial eprom.

fedplxeper: WARNING: This command erases contents of PLX serial eprom
fedplxeper: Do you really want to do this?
fedplxeper: Type 'yes' to continue any other character to abort =>
fedplxeper: OK continuing...
********
ERASING PLX Eprom...
FEDPMC_DEBUG > Erased plx eprom done.
PLX serial eprom is erased.
FED_Mon>
```

See also

fedplxepfl, fedplxeprd, fedplxepwr

9.28 FEDPLXEPFL

Name

fedplxepfl: Fill contents of PLX serial eprom with default values.

Syntax

fedplxepfl <address> <data>

Parameters

< address > short word location in serial eprom 0-63 {$0}
< data>16-bit valuel{ $0000}

Description

WARNING: This command is for Experts only.

Overwrites the contents of the serial eprom attached to PLX PCI bridge chip with default values.
Eprom contains 64 x 16 bit locations.

Example

Fill contents of PLX serial eprom with default values..

```
FED_Mon>fedplxepfl
fedplxepfl:  Fill contents of PLX serial eprom with default values.

fedplxepfl: WARNING: This command overwrites contents of PLX serial
eprom with default values
fedplxepfl: Do you really want to do this?
fedplxepfl: Type 'yes' to continue any other character to abort =>
fedplxepfl: OK continuing...
********
Writing to PLX Eprom...
Starting at location 0; for 64 locations
Eprom word # 00 ; wrote $fed0
Eprom word # 01 ; wrote $10dc
Eprom word # 02 ; wrote $0000
Eprom word # 03 ; wrote $0000
Eprom word # 04 ; wrote $0000
Eprom word # 05 ; wrote $0000
Eprom word # 06 ; wrote $0000
Eprom word # 07 ; wrote $0000
Eprom word # 08 ; wrote $0000
Eprom word # 09 ; wrote $0000
Eprom word # 10 ; wrote $fff0
Eprom word # 11 ; wrote $0000
Eprom word # 12 ; wrote $0000
Eprom word # 13 ; wrote $0001
Eprom word # 14 ; wrote $0000
Eprom word # 15 ; wrote $0000
Eprom word # 16 ; wrote $0000
Eprom word # 17 ; wrote $0000
Eprom word # 18 ; wrote $0000
Eprom word # 19 ; wrote $0000
Eprom word # 20 ; wrote $0000
Eprom word # 21 ; wrote $0000
Eprom word # 22 ; wrote $0303
Eprom word # 23 ; wrote $00c3
Eprom word # 24 ; wrote $0000
Eprom word # 25 ; wrote $0000
Eprom word # 26 ; wrote $0000
Eprom word # 27 ; wrote $0000
Eprom word # 28 ; wrote $0000
Eprom word # 29 ; wrote $0000
Eprom word # 30 ; wrote $0000
Eprom word # 31 ; wrote $0000
Eprom word # 32 ; wrote $0000
Eprom word # 33 ; wrote $0000
Eprom word # 34 ; wrote $9080
Eprom word # 35 ; wrote $10b5
Eprom word # 36 ; wrote $ffff
Eprom word # 37 ; wrote $ff00
Eprom word # 38 ; wrote $0010
Eprom word # 39 ; wrote $0001
Eprom word # 40 ; wrote $0000
Eprom word # 41 ; wrote $0243
Eprom word # 42 ; wrote $0000
Eprom word # 43 ; wrote $0000
Eprom word # 44 ; wrote $0000
Eprom word # 45 ; wrote $0000
Eprom word # 46 ; wrote $0000
Eprom word # 47 ; wrote $0000
Eprom word # 48 ; wrote $0000
Eprom word # 49 ; wrote $0000
Eprom word # 50 ; wrote $0000
Eprom word # 51 ; wrote $0000
Eprom word # 52 ; wrote $0000
Eprom word # 53 ; wrote $0000
Eprom word # 54 ; wrote $0000
Eprom word # 55 ; wrote $0000
Eprom word # 56 ; wrote $0000
Eprom word # 57 ; wrote $0000
Eprom word # 58 ; wrote $0000
Eprom word # 59 ; wrote $0000
Eprom word # 60 ; wrote $0000
Eprom word # 61 ; wrote $0000
Eprom word # 62 ; wrote $0000
Eprom word # 63 ; wrote $0000
PLX serial eprom is reprogrammed with default values.
FED_Mon>
```

See also

fedplxeper, fedplxeprd, fedplxepwr

9.29 FEDPLXEPRD

Name

fedplxeprd: Read contents of PLX serial eprom.

Syntax

fedplxeprd

Parameters

none

Description

Display the contents of serial eprom attached to PLX PCI bridge chip.
Contents read 0xFFFF if eprom is not present on PMC or if it is erased.
Eprom contains 64 x 16 bit locations.

Example

Display PLX eprom contents (following list shows default values).

```
FED_Mon>fedplxeprd
********
Reading from PLX Eprom...
PLX serial eprom contents:
location 00 = 0xfed0
location 01 = 0x10dc
location 02 = 0x0000
location 03 = 0x0000
location 04 = 0x0000
location 05 = 0x0000
location 06 = 0x0000
location 07 = 0x0000
location 08 = 0x0000
location 09 = 0x0000
location 10 = 0xfff0
location 11 = 0x0000
location 12 = 0x0000
location 13 = 0x0001
location 14 = 0x0000
location 15 = 0x0000
location 16 = 0x0000
location 17 = 0x0000
location 18 = 0x0000
location 19 = 0x0000
location 20 = 0x0000
location 21 = 0x0000
location 22 = 0x0303
location 23 = 0x00c3
location 24 = 0x0000
location 25 = 0x0000
location 26 = 0x0000
location 27 = 0x0000
location 28 = 0x0000
location 29 = 0x0000
location 30 = 0x0000
location 31 = 0x0000
location 32 = 0x0000
location 33 = 0x0000
location 34 = 0x9080
location 35 = 0x10b5
location 36 = 0xffff
location 37 = 0xff00
location 38 = 0x0010
location 39 = 0x0001
location 40 = 0x0000
location 41 = 0x0243
location 42 = 0x0000
location 43 = 0x0000
location 44 = 0x0000
location 45 = 0x0000
location 46 = 0x0000
location 47 = 0x0000
location 48 = 0x0000
location 49 = 0x0000
location 50 = 0x0000
location 51 = 0x0000
location 52 = 0x0000
location 53 = 0x0000
location 54 = 0x0000
location 55 = 0x0000
location 56 = 0x0000
location 57 = 0x0000
location 58 = 0x0000
location 59 = 0x0000
location 60 = 0x0000
location 61 = 0x0000
location 62 = 0x0000
location 63 = 0x0000
FED_Mon>
```

See also

fedplxeper, fedplxepfl, fedplxepwr

9.30 FEDPLXEPWR

Name

fedplxepwr: Write to an individual location in PLX serial eprom.

Syntax

fedplxepwr

Parameters

none

Description

WARNING: This command is for Experts only.

Overwrites the contents of a single 16 bit location in the serial eprom attached to PLX PCI bridge chip with specified value.
1st location is at address 0.

Example

Fill contents of 2nd location in PLX serial eprom with value = $beef.

```
FED_Mon>fedplxepwr 1 beef
fedplxepwr: Write to an individual location in PLX serial eprom.
fedplxepwr: WARNING: This command overwrites contents of PLX serial
eprom.
fedplxepwr: Do you really want to do this?
fedplxepwr: Type 'yes' to continue any other character to abort =>
fedplxepwr: OK continuing...
********
Writing to PLX Eprom...
Starting at location 1; for 1 locations
Eprom word # 01 ; wrote $beef
PLX serial eprom is reprogrammed with new value.
FED_Mon>
```

See also

fedplxeper, fedplxepfl, fedplxeprd

9.31 FEDREV

Name

fedrev – display FED hardware and software revision information

Syntax

fedrev

Parameters

none

Description

Show current hardware and software revisions.

Example

Show revisions.

```
FED_Mon>fedrev
fedrev:  Version information :

FIRMWARE:
Xilinx : version = 0
       : revison = 2
       : prototype = 0

SOFTWARE:
FED_Mon Version of Dec  6 2000 at 11:05:34
FEDPMC s/w library version = '2.07'
FED_Mon>
```

See also

fedidrd

9.32 FEDROUT

Name

fedrout – Readout and display FED next pending event in DPM.

Syntax

fedrout
fedrout <dest_address >< format_flag > <dma_flag > < display_format >

Parameters

< dest_address > data destination in RIO memory {$0030'0000}
< format_flag > FED data format { 1=NO Formatting }
< dma_flag > {0=OFF ; 1=ON}
< format_flag > display format {1=hex ; 2=decimal}

Description

Read out the next pending event (if present) in DPM. It performs the commands necessary for FED readout sequence and displays event data and associated fifo contents (i.e. event & bunch crossing numbers).
If no arguments are passed defaults are used.

Example

Readout first event pending in DPM.

```
FED_Mon>fedstart 1
fedstart:  FED  data  acquisition  Run  is  STARTED,  using  SOFTWARE
triggers.

FED_Mon>fedswt 4
fedswt: sent software trigger #1.
fedswt: sent software trigger #2.
fedswt: sent software trigger #3.
fedswt: sent software trigger #4.
fedbuf: number of filled buffers = 4.

FED_Mon>fedrout
readout time for 16 samples (skip begin/end = 0/0) = 452 micro sec
fedrout: Readout Event data to RIO address $00300000

Event no = $00000000 ; Bunch no = 49148
Event size = 256 bytes; Number Samples = 16; Skipped samples begin/end
= 0/0;
Buffer base in dpm = $00000000

Unformatted data follows:

ADC Chan# =>  0    1    2    3    4    5    6    7
Sample #
       000:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
       001:  0284 0278 0274 0277 0280 0280 0281 0279   (Dec)
       002:  0284 0279 0274 0277 0280 0280 0280 0279   (Dec)
       003:  0284 0278 0274 0277 0280 0279 0280 0278   (Dec)
       004:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
       005:  0284 0278 0275 0277 0280 0280 0280 0279   (Dec)
       006:  0284 0279 0274 0277 0280 0280 0281 0279   (Dec)
       007:  0284 0279 0275 0277 0280 0280 0281 0279   (Dec)
```

```
               008:  0284 0279 0274 0277 0280 0280 0280 0279   (Dec)
               009:  0284 0279 0274 0277 0280 0280 0281 0279   (Dec)
               010:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
               011:  0284 0278 0274 0277 0280 0280 0280 0279   (Dec)
               012:  0284 0278 0274 0277 0280 0280 0280 0278   (Dec)
               013:  0284 0278 0274 0277 0280 0280 0280 0278   (Dec)
               014:  0284 0279 0275 0277 0280 0280 0281 0279   (Dec)
               015:  0284 0279 0274 0277 0280 0280 0281 0279   (Dec)

Number of events pending = 3
FED_Mon>
```

See also

fedstart, fedswt

9.33 FEDSCP

Name

fedscp – Scope mode test reads, writes or toggles a single FED memory address.

Syntax

fedscp
fedscp <address_offset><operation>

Parameters

< address_offset > test address offset from FED memory base { $0 }
< operation > { 1=Read ; 2=Write ; 3=Toggle}

Description

Continuously reads, writes or toggles a specified memory address.
Useful for analysing lines on the scope.
If no arguments are passed the first location in the DPM is filled.
Actions are terminated by hitting any character on the keyboard.

Example

Read long word continuously from first location in DPM.

```
FED_Mon>fedscp 0 1
fedscp:  CONTINUOUSLY READING @ addr = $c0000000; type any character
to stop...
fedscp: stopped..
FED_Mon>
```

See also

feddp, fedfp

9.34 FEDSTART

Name

fedstart – Starts a data acquisition run.

Syntax

fedstart
fedstart<trigger_type>

Parameters

< trigger_type > { 0=HW Triggers ; 1=SW Triggers }

Description

Starts a data acquistion run by enabling ADC digitisation.

Example

Start run with software triggers.

```
FED_Mon>fedstart 1
fedstart: FED data acquisition Run is STARTED, using SOFTWARE
triggers.
FED_Mon>
```

See also

fedstop, fedrout

9.35 FEDSTAT

Name

fedstat – display FED PMC status

Syntax

fedstat
fedstat <option>

Parameters

< option >

b	show bridge status only
r	show fed registers status only

Description

Display status of FED PMC.
If no arguments are passed full settings are displayed.

Example

Display full settings of FED PMC.

```
FED_Mon>fedstat

fedstat: fed pmc full status:-

bridge config regs @ pci cfg base = $80802000

                 vendor id = $908010b5
     bridge cmd/status reg = $02800007
    dpm pci mem base (rio) = $c0000000
   fifo pci mem base (rio) = $c0100000
 bridge pci mem base (rio) = $c0100100

 bridge local regs: @ pci mem base = $c0100100

      dpm local base = $00000001
     dpm window size = $fff00000
      dpm descriptor = $000301c3
     regs local base = $00100001
    regs window size = $ffffff00
     regs descriptor = $00000243
  mode & arbitration = $01200000
        eeprom cntrl = $88000000
          big endian = $00000004

fedstat: ATTENTION => If we stop here the Xilinx is NOT Loaded.
fedstat: reset default bridge configuration.
fedstat: reset default bridge configuration.
 fed regs: @ pci mem base = $c0100080

 Digitisation is >> DISABLED <<

    Test mode is << ENABLED >>

     Throttle is >> DISABLED <<
 throttle threshold = $000000ab
```

```
      Clock source = 0
      Clock source => PCI CLOCK
       Clock delay = 0

    Trigger source = 0
    Trigger source => FRONT PANEL TRIGGER
      Trigger mode = 0
      Trigger mode => START DIGITISATION

    adc_chan_mask = $aa
    ADC chans 0&1 are << ENABLED >>
              2&3 are << ENABLED >>
              4&5 are << ENABLED >>
              6&7 are << ENABLED >>
    ADC Sample Freq = 0
    ADC Sample Size = 256

    Number of Filled Buffers = 7

  FED_Mon>
```

See also

fedinit

9.36 FEDSTOP

Name

fedstop – Stops a data acquisition run.

Syntax

fedstop

Parameters

none

Description

Stops a data acquistion run by disabling ADC digitisation.

Example

Stop run.

```
FED_Mon>fedstart 1
fedstart:  FED  data  acquisition  Run  is  STARTED,  using  SOFTWARE
triggers.

FED_Mon>fedstop
fedstop: FED data acquisition Run is STOPPED.
FED_Mon>
```

See also

fedstart

9.37 FEDSWT

Name

fedswt – sends software triggers.

Syntax

fedswt
fedswt [<number_triggers>]

Parameters

< number_triggers > number of software triggers to generate

Description

Send software triggers
If no arguments are passed a single trigger is sent.

Example

Send 4 software triggers.

```
FED_Mon>fedswt 4
fedswt: sent software trigger #1.
fedswt: sent software trigger #2.
fedswt: sent software trigger #3.
fedswt: sent software trigger #4.

fedswt: number of filled buffers = 4.
FED_Mon>
```

See also

9.38 FEDTEST

Name

fedtest – test DPM contents.

Syntax

fedtest
fedtest <address_offset><number_reads> <max_errors>

Parameters

< address_offset > start test at offset from FED memory base {$0 }
< number_locations > number of locations (longs) to test { 2 }
<max_errors> stop after number of bad locations { 1 }

Description

Tests DPM contents with a set of standard patterns and reports any errors found.
If no arguments are passed defaults are used.

Example

Test first 16 locations in DPM. Stop if any locations show errors.

```
FED_Mon>fedtest 0 16 1
fedtest: begin @ addr = $00000000; number locations = 22 (will stop
after 1 bad locations)...;
fedtest: tested location in dpm at offset @ $00000000 ; bad locations
so far = 0

fedtest: stopped @ addr = $c0000058; #locations = 22; #bad locs = 0
#errors = 0;
FED_Mon>
```

See also

9.39 FEDTHR

Name

fedthr – display/set FED Trigger Throttle mode

Syntax

fedthr
fedthr [<operation>]

Parameters

< operation>

0	Disable Trigger Throttle mode
1	Enable Trigger Throttle mode

Description

Set/Display Trigger Throttle mode.

Example

Enable Trigger Throttle.

```
FED_Mon>fedthr 1
fedthr: Throttle mode is << ENABLED >>
FED_Mon>
```

See also

9.40 FEDTMOD

Name

fedtmod– display/set FED Test mode

Syntax

fedtmod
fedtmod [<operation>]

Parameters

< operation>

0	Disable FED Test mode
1	Enable FED Test mode

Description

Enable or Disable Test mode.
If no parameter is passed display status.

Example

Enable test mode.

```
FED_Mon>fedtmod 1
fedtmod: Test mode is << ENABLED >>
FED_Mon>
```

See also

9.41 FEDTRG

Name

fedtrg – display/set External trigger source and mode

Syntax

fedtrg
fedtrg <trigger_source><trigger_mode>

Parameters

< trigger_source > {0=Front Panel;2=J4 LVDS;3=J4 TTL}
< trigger_mode > {0=Start Digitisation}

Description

Set/Display External trigger source and mode.
If no arguments are passed the trigger status is displayed.
NB fedswt can be used to generate software triggers.

Example

Select front panel external triggers.

```
FED_Mon>fedtrg 0 0
fedtrg: trigger source = FRONT PANEL TRIGGER
fedtrg:   trigger mode = START DIGITISATION
FED_Mon>
```

See also

fedswt

9.42 FEDXLST

Name

fedxlst – Read Xilinx load status.

Syntax

fedxlst

Parameters

none

Description

Reads the Xilinx load status via the Lattice serial line.

Example

Read Xilinx load status.

```
FED_Mon>fedxlst
fedxlst: Xilinx Load status : INIT = 1, DONE = 1
FED_Mon>
```

See also

9.43 FEDAPVCTRD *Firmware v2 only*

Name

fedapvctrd – Display APV frame and trigger counters.

Syntax

fedapvctrd

Parameters

none

Description

Display values of Frame and Trigger counters.
cf Event and BX counters read from external FIFO.

Example

Display APV counters.

```
FED_Mon>fedapvctrd
fedapvctrd:   APV Frame Counter = 0 (dec).
fedapvctrd: APV Trigger Counter = 0 (dec).
FED_Mon>
```

See also

fedapvctrs

9.44 FEDAPVCTRS *Firmware v2 only*

Name

fedapvctrs – Reset APV frame and trigger counters.

Syntax

fedapvctrs

Parameters

none

Description

Reset APV frame and trigger counters.
cf Event and BX counters read from external FIFO.

Example

Reset APV frame and trigger counters.

```
FED_Mon>fedapvctrs
fedapvctrs:  APV Frame & Trigger Counters Reset.
FED_Mon>
```

See also

fedapvctrd

9.45 FEDAPVMOD *Firmware v2 only*

Name

fedapvmod– display/set FED APV sync mode

Syntax

fedapvmod
fedapvmod [<operation>]

Parameters

< operation>

0	Disable FED APV sync mode
1	Enable FED APV sync mode

Description

Enable or Disable APV sync mode.
If no parameter is passed display status.

Nb APV sync mode must be enabled for BOTH Header Finding AND New scope mode.

Example

Enable APV sync mode.

```
FED_Mon>fedapvmod 1
fedapvmod: APV Sync mode is << ENABLED >>
FED_Mon>
```

See also

fedapvscp, fedapvthr

9.46 FEDAPVPARM *Firmware v2 only*

Name

fedapvparm – Display/Set APV timeout parameters override values.

Syntax

fedapvparm
fedapvparm [< frame_timeout> < tick_timeout> <frame_size>]

Parameters

< frame_timeout > APV frame timeout ovveride
< tick_timeout > APV tick timeout ovveride
< frame_size > APV frame_size ovveride

Description

Set APV timeout parameters ovveride values.
If no arguments are passed timeout parameter override values are displayed.

APV Timeout parameters control details of header finding and frame capture.

APV Timeout parameters override mode must be enabled with fedapvpov to enable the new values.

If APV Timeout parameters override mode is Disabled following reset defaults are used:
frame_timeout = 275
tick_timeout = 65
frame_size = 277

Example

Set APV timeout parameters override values for sampling at 1/2 clock frequency.

```
FED_Mon>fedapvparm 0d275 0d65 0d139
fedapvparm: Set/Read APV timeout parameters.
fedapvparm: APV frame parms: frame_timeout = 0d275 ; tick_timeout =
0d65 ; frame_size = 0d139 .
FED_Mon>
```

See also

fedapvpov

9.47 FEDAPVPOV ***Firmware v2 only***

Name

fedapvpov– display/set FED APV Timeout parameters override mode.

Syntax

fedapvpov
fedapvpov [<operation>]

Parameters

< operation>

0	Disable APV Timeout parameters override
1	Enable APV Timeout parameters override

Description

Enable or Disable APV Timeout parameters override.
If no parameter is passed display status.

APV Timeouts must be overridden before new values set by fedapvparm become active.

If APV Timeout parameters override mode is Disabled following reset defaults are used.

Example

Enable APV Timeout parameters override.

```
FED_Mon>fedapvpov 1
fedapvpov: APV Timeout parameters override is << ENABLED >>
FED_Mon>
```

See also

fedapvparm

9.48 FEDAPVSCP *Firmware v2 only*

Name

fedapvscp– display/set FED APV new Scope mode

Syntax

fedapvscp
fedapvscp [<operation>]

Parameters

< operation>

0	Disable FED APV new Scope mode
1	Enable FED APV new Scope mode

Description

Enable or Disable APV new Scope mode.
If no parameter is passed display status.

Nb APV new Scope mode must be DISABLED for APV header finding.

Example

Enable APV new Scope mode.

```
FED_Mon>fedapvscp 1
fedapvscp: APV (NEW) Scope mode is << ENABLED >>
FED_Mon>
```

See also

fedapvmod

9.49 FEDAPVTHR *Firmware v2 only*

Name

fedapvthr – Display/Set APV Header thresholds.

Syntax

fedapvthr
fedapvthr [<high_threshold> <low_threshold>]

Parameters

< high_threshold > APV header threshold high
< low_threshold > APV header threshold low

Description

Set high and low thresholds for APV header finding.
If no arguments are passed threshold values are displayed.

Example

Set high threshold = 100 and low threshold = 50.

```
FED_Mon>fedapvthr 0d100 0d50
fedapvthr: Set APV header thresholds.
fedapvthr: APV frame: thresh_high = 0d100 ; thresh_low = 0d50
FED_Mon>
```

See also

9.50 FEDFEND *Firmware v2 only*

Name

fedfend– display/set FED FrontEnd ADCs to DPM mode

Syntax

fedfend
fedfend [<operation>]

Parameters

< operation>

0	Disable FrontEnd ADCs to DPM
1	Enable FrontEnd ADCs to DPM

Description

Enable or Disable FrontEnd ADCs to DPM.
If no parameter is passed display status.

The FrontEnd ADCs to DPM can be disabled ie BYPASSED for pattern generation tests to DAQ.
Normally FrontEnd ADCs to DPM should be enabled for ADC capture.

Example

Enable FrontEnd ADCs to DPM.

```
FED_Mon>fedfend 1
fedfend: Frontend is << ENABLED >>
FED_Mon>
```

See also

feddpmfl

9.51 FEDINTFIFO *Firmware v2 only*

Name

fedintfifo– display/enable FED Internal FIFO.

Syntax

fedintfifo
fedintfifo [<operation>]

Parameters

< operation>

0	Disable FED Internal FIFO
1	Enable FED Internal FIFO

Description

Enable or Disable Internal FIFO.
If no parameter is passed display status.

Internal FIFO is needed to handle triggers with separation < 7 BX.

Example

Enable Internal FIFO.

```
FED_Mon>fedintfifo 1
fedintfifo: Internal FIFO is << ENABLED >>
FED_Mon>
```

See also

9.52 FEDTRGFLT *Firmware v2 only*

Name

fedtrgflt– display/enable FED Trigger filter.

Syntax

fedtrgflt
fedtrgflt [<operation>]

Parameters

< operation>

0	Disable FED Trigger Filter
1	Enable FED Trigger Filter

Description

Enable or Disable Trigger Filter.
If no parameter is passed display status.

Filter rejects External Triggers with illegal patterns eg '101'.

Example

Enable Trigger Filter.

```
FED_Mon>fedtrgflt 1
fedtrgflt: Trigger Filter is << ENABLED >>
FED_Mon>
```

See also

10 Document Servers

Latest versions of all FED PMC documentation and software are available from the following web site:

hepwww.rl.ac.uk/CMS_fed/Default.htm

or alternatively directly at the following anonymous ftp server:

address : **ftp.te.rl.ac.uk**
directory: **cms/fed/fed_pmc/**

11 References

[1]	FED-PMC User Manual	ftp://ftp.te.rl.ac.uk/cms/fed/docs/manuals/fedpmc_um.pdf
[2]	CES RIO8062	http://www.ces.ch/Products/Products.html
[3]	CES AWX 3317C	http://www.ces.ch/Products/Products.html